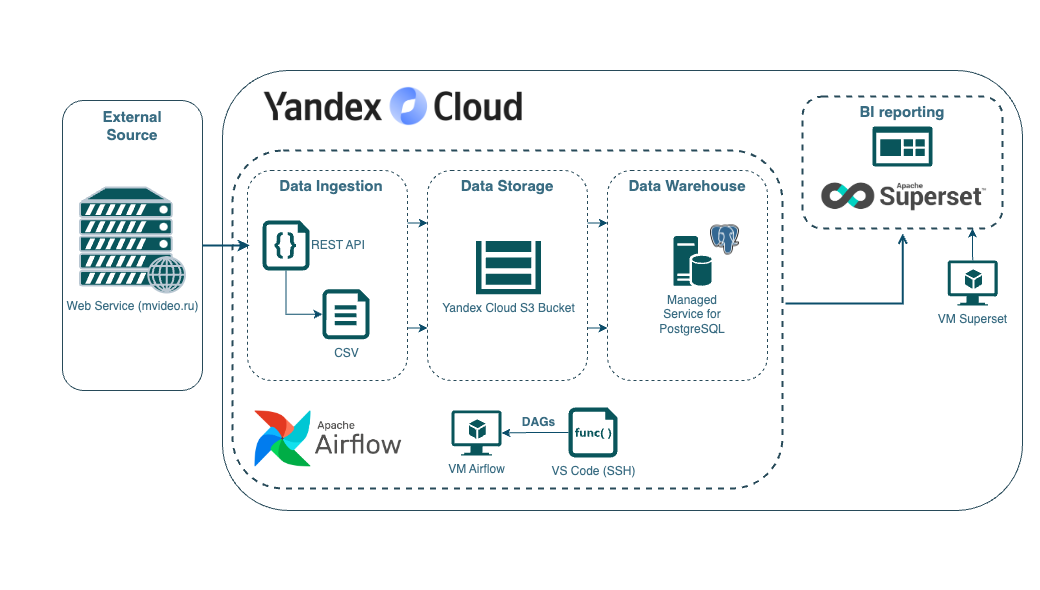
Шейко Алина

OTUS

**Проектная работа на тему “Анализ рынка техники на базе web scraping и Apache Superset”**

Целью работы являлось проанализировать рынок современной техники на основе информации с веб-сайта Mvideo при помощи инструментов в инфраструктуре Yandex Cloud.

Архитектурное решение проекта выбрано следующее:



Ссылка на репозиторий Github: <https://github.com/AlinaSheik05/otus_project>

**Части проекта:**

1. Сбор данных
2. Обработка данных
3. Загрузка данных в Data Lake
4. Загрузка данных в DWH
5. Формирование витрины данных
6. Построение аналитической отчетности на основе витрины данных

**Сбор данных, Обработка данных**

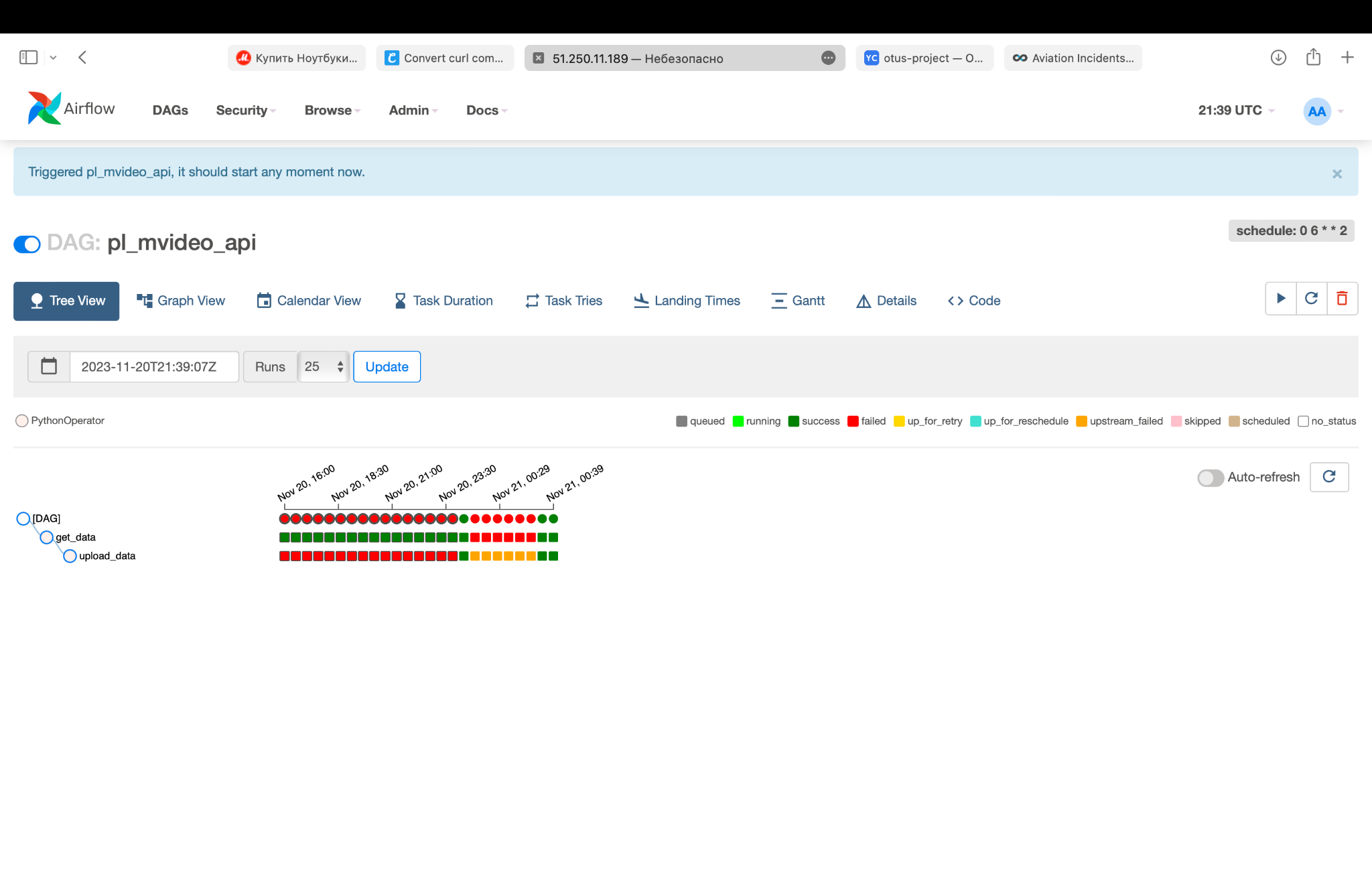
Для начала было создано облако в Yandex Cloud.

С целью сбора и обработки данных был развернут виртуальный сервер на базе Ubuntu, на который был установлен Apache Airflow с помощью Docker-контейнера.

Все настройки сервера, а также установка необходимых библиотек происходили через Yandex CLI.

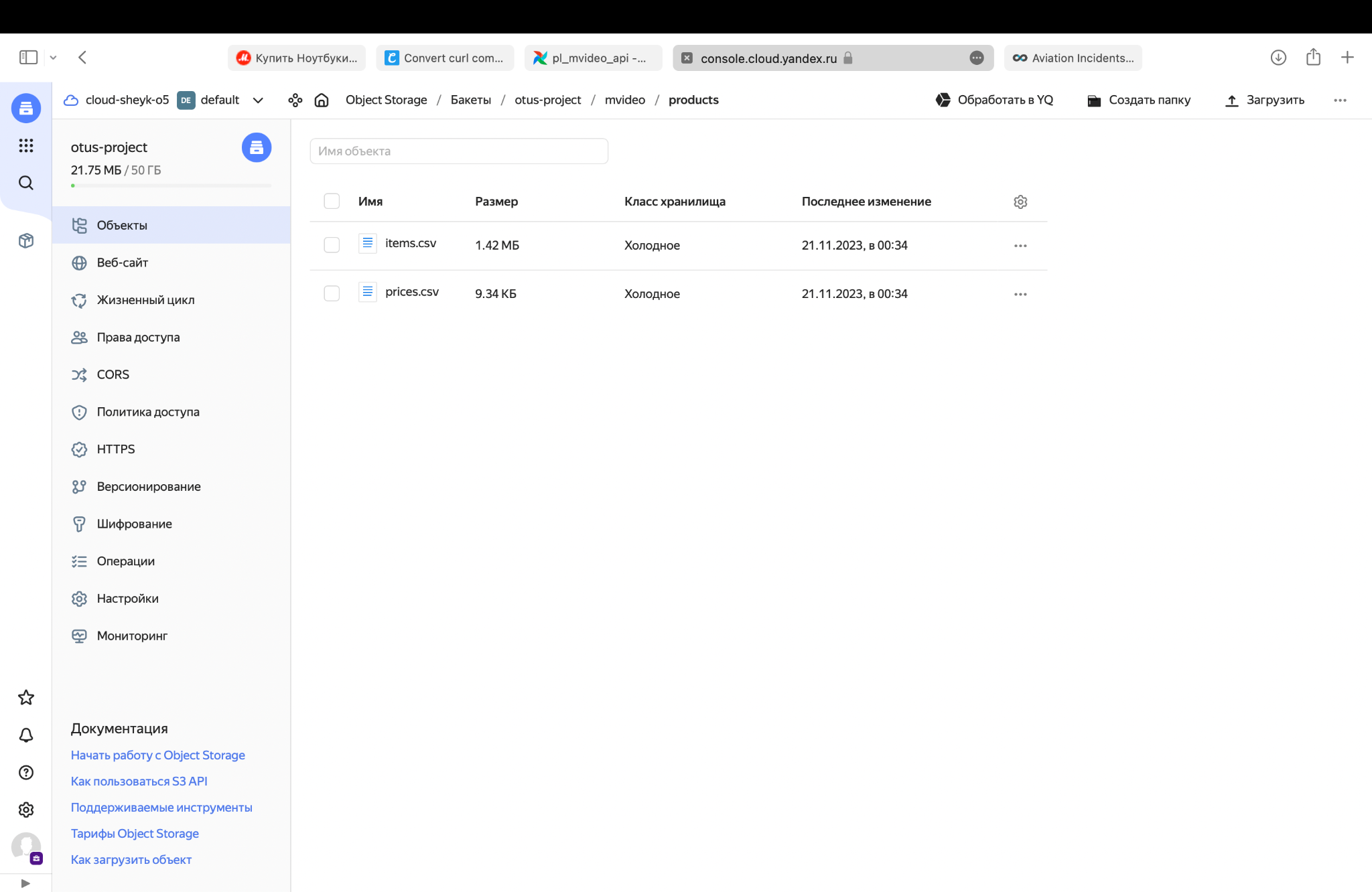
Далее с помощью подключения к серверу по SSH в Visual Studio Code были написаны пайплайны загрузки данных (DAGs) с настроенным расписанием на ежедневное обновление.

Подключение к веб-серверу mvideo.ru реализовано на основе REST API запросов. Полученные json файлы преобразованы в датафрейм. (Код в Приложении)



**Загрузка данных в Data Lake**

Далее датафреймы были преобразованы в .csv формат и загружены в предварительно созданный бакет Yandex Object Storage с целью хранения сырых данных.



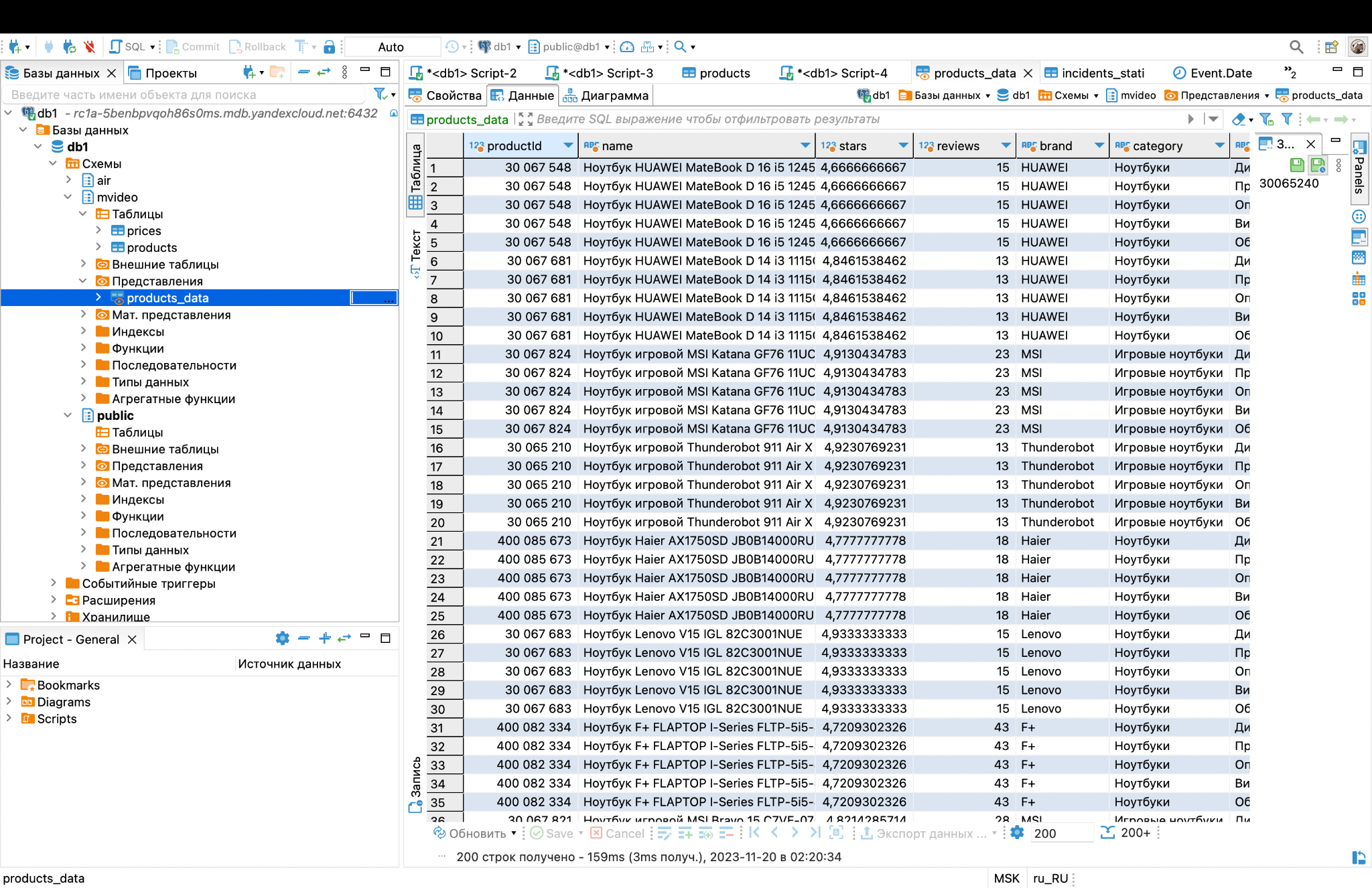
**Загрузка данных в DWH**

При помощи сервиса Managed Service for PostgreSQL было развернуто аналитическое хранилище данных.

Пайплайн Airflow предопределяет типы данных и загружает данные в хранилище.

**Формирование витрины данных**

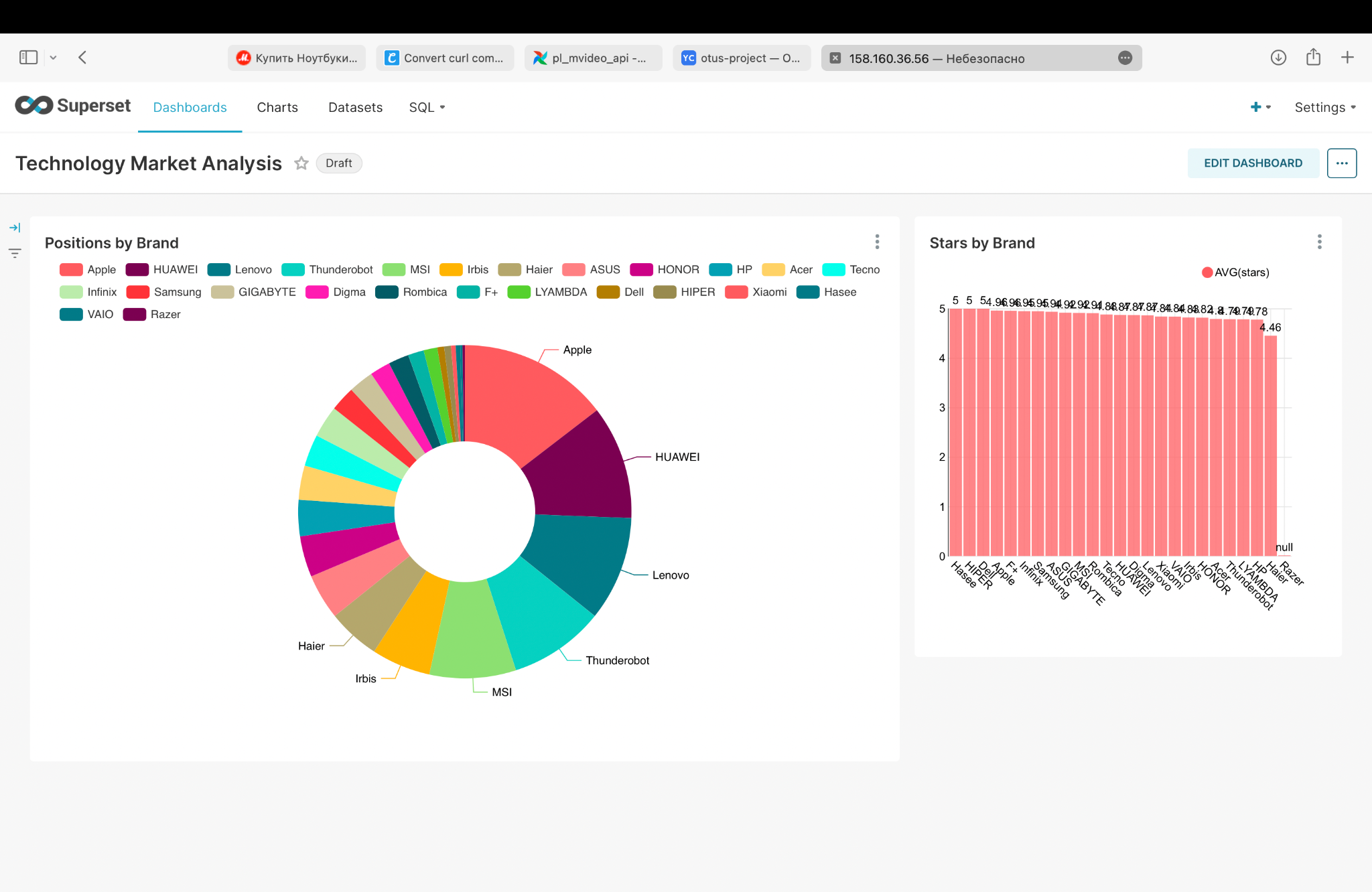
На загруженных таблицах сформирована витрина в формате view.



**Построение аналитической отчетности на основе витрины данных**

Развернут виртуальный сервер на базе Ubuntu, где также развернут Apache Superset при помощи Docker-контейнера.

Созданы подключение к БД PostgreSQL, датасет на сформированной витрине, а также чарты с расчетами необходимых метрик.  
На основе созданных чартов собран дашборд.



ПРИЛОЖЕНИЕ

Код DAG в Airflow:

import logging as \_log

from airflow import DAG

from airflow.utils.dates import days\_ago

from airflow.providers.postgres.hooks.postgres import PostgresHook

from airflow.hooks.base import BaseHook

from airflow.hooks.S3\_hook import S3Hook

from airflow.operators.python import PythonOperator

from airflow.providers.postgres.operators.postgres import PostgresOperator

from datetime import datetime

from io import StringIO, BytesIO

import boto3

import requests

import json

import pandas as pd

import sqlalchemy

from sqlalchemy import create\_engine

*def* get\_data():

#Variable.get("obs\_keys")

cookies = {

'\_ga\_BNX5WPP3YK': 'GS1.1.1700336107.7.0.1700336107.60.0.0',

'\_ga\_CFMZTSS5FM': 'GS1.1.1700336107.7.0.1700336107.0.0.0',

'MVID\_CITY\_ID': 'CityCZ\_975',

'MVID\_KLADR\_ID': '7700000000000',

'MVID\_REGION\_ID': '1',

'MVID\_REGION\_SHOP': 'S002',

'MVID\_TIMEZONE\_OFFSET': '3',

'MVID\_ENVCLOUD': 'prod2',

'\_\_hash\_': '2fbf022f890c5be747609d06b1360a46',

'fgsscgib-w-mvideo': 's5izae8a324fe3a1120e5b8a91060ac5fdd48873',

'gsscgib-w-mvideo': 'hUc/SVSGgmztX3Of0oeximZoUbC8rJzsseLHCjxrtjK596Dw0/6wQnZECkUgxNqaJ+Y8g0DKz9x3/U+sldGU+a46wHDoOG6wNs0mAXwMXRfksnyL9wxWdbB50WAvGRMibtljNstsc7xlcxa/bsrsl0DEc9O6D8Hs4wGk9CgFplY6hMSxQ54oLneQFG82YQe2OHZQfSnxgpNBmu3YYkp/P9/D6i3BsS2cwi96WC9rzXKunuLjiuwS9SAfhl5ueA==',

'fgsscgib-w-mvideo': 's5izae8a324fe3a1120e5b8a91060ac5fdd48873',

'gsscgib-w-mvideo': 'hUc/SVSGgmztX3Of0oeximZoUbC8rJzsseLHCjxrtjK596Dw0/6wQnZECkUgxNqaJ+Y8g0DKz9x3/U+sldGU+a46wHDoOG6wNs0mAXwMXRfksnyL9wxWdbB50WAvGRMibtljNstsc7xlcxa/bsrsl0DEc9O6D8Hs4wGk9CgFplY6hMSxQ54oLneQFG82YQe2OHZQfSnxgpNBmu3YYkp/P9/D6i3BsS2cwi96WC9rzXKunuLjiuwS9SAfhl5ueA==',

'cfidsgib-w-mvideo': 'pp5YhKczZl+Am+zuOzYAY4TARryMcS7lEwri/lsp5RsYyIN0pMHMEsNUOnGa4bLUqN6rGfXb0sBA77d299X91sh70MP95DYmeFzHlzqJnfnerPVKa+gJw/C691qmBYZBSokDpU9p6ugLMw9X0M+VLvNHsWBBi69DdVb5XJ4=',

'tmr\_detect': '0%7C1700260968223',

'afUserId': '659a4d1e-af64-4801-8fff-85a6819abcb1-p',

'tmr\_lvid': '6d42014a97f2726be06d238fe1bc121f',

'tmr\_lvidTS': '1700242055898',

'\_sp\_id.d61c': '4794ea78-98eb-47ee-8743-871b2a0e2c51.1700242052.6.1700260963.1700259010.8d1d8e58-fc1e-461e-b8fc-ba0fa74c4aca.c0f7db55-9246-4023-ac2f-82782f0fe8a0.fdf2d5be-8eed-49a7-b27d-cc6f15288d8d.1700260962200.11',

'\_ga': 'GA1.1.1178978048.1700242053',

'directCrm-session': '%7B%22deviceGuid%22%3A%221986d968-c4ea-4164-a67d-e050dc78cab3%22%7D',

'mindboxDeviceUUID': '1986d968-c4ea-4164-a67d-e050dc78cab3',

'CACHE\_INDICATOR': 'true',

'COMPARISON\_INDICATOR': 'false',

'MVID\_NEW\_OLD': 'eyJjYXJ0IjpmYWxzZSwiZmF2b3JpdGUiOnRydWUsImNvbXBhcmlzb24iOnRydWV9',

'MVID\_VIEWED\_PRODUCTS': '',

'bIPs': '1613809182',

'flacktory': 'no',

'BIGipServeratg-ps-prod\_tcp80': '1694817290.20480.0000',

'BIGipServeratg-ps-prod\_tcp80\_clone': '1694817290.20480.0000',

'SMSError': '',

'authError': '',

'deviceType': 'desktop',

'MVID\_GTM\_BROWSER\_THEME': '1',

'HINTS\_FIO\_COOKIE\_NAME': '1',

'JSESSIONID': 'D2JnlXTSMhjhmD2CSL03L6VzC1sMgF7Q8T9gKZpqsVnh6SVml5nN!14028060',

'MVID\_CALC\_BONUS\_RUBLES\_PROFIT': 'true',

'MVID\_CART\_MULTI\_DELETE': 'true',

'MVID\_GET\_LOCATION\_BY\_DADATA': 'DaData',

'MVID\_GUEST\_ID': '23221042682',

'MVID\_OLD\_NEW': 'eyJjb21wYXJpc29uIjogdHJ1ZSwgImZhdm9yaXRlIjogdHJ1ZSwgImNhcnQiOiB0cnVlfQ==',

'MVID\_YANDEX\_WIDGET': 'true',

'NEED\_REQUIRE\_APPLY\_DISCOUNT': 'true',

'PRESELECT\_COURIER\_DELIVERY\_FOR\_KBT': 'false',

'PROMOLISTING\_WITHOUT\_STOCK\_AB\_TEST': '2',

'searchType2': '2',

'wurfl\_device\_id': 'generic\_web\_browser',

'AF\_SYNC': '1700242248615',

'adrcid': 'A2jKFL8Zktj74inloSi6mqQ',

'\_gpVisits': '{"isFirstVisitDomain":true,"idContainer":"100025D5"}',

'advcake\_session\_id': '011376e3-f880-786e-d70b-6e60b24956f6',

'advcake\_track\_id': 'fbd92231-a646-9114-e887-3b849ce9a125',

'gdeslon.ru.\_\_arc\_domain': 'gdeslon.ru',

'gdeslon.ru.user\_id': '23ab5141-b9f1-4388-8038-92cb1604dc1d',

'uxs\_uid': '99357be0-856e-11ee-8157-bfca0099a8c8',

'flocktory-uuid': 'e4fef67c-a2ad-4a02-b185-a46d9b936ee3-4',

'\_ym\_d': '1700242053',

'\_ym\_uid': '1700242053847633239',

'MVID\_AB\_PERSONAL\_RECOMMENDS': 'true',

'MVID\_AB\_UPSALE': 'true',

'MVID\_ALFA\_PODELI\_NEW': 'true',

'MVID\_CASCADE\_CMN': 'true',

'MVID\_CHAT\_VERSION': '4.16.4',

'MVID\_CREDIT\_DIGITAL': 'true',

'MVID\_CREDIT\_SERVICES': 'true',

'MVID\_CRITICAL\_GTM\_INIT\_DELAY': '3000',

'MVID\_CROSS\_POLLINATION': 'true',

'MVID\_DISPLAY\_ACCRUED\_BR': 'true',

'MVID\_EMPLOYEE\_DISCOUNT': 'true',

'MVID\_FILTER\_CODES': 'true',

'MVID\_FILTER\_TOOLTIP': '1',

'MVID\_FLOCKTORY\_ON': 'true',

'MVID\_GTM\_ENABLED': '011',

'MVID\_INTERVAL\_DELIVERY': 'true',

'MVID\_IS\_NEW\_BR\_WIDGET': 'true',

'MVID\_LAYOUT\_TYPE': '1',

'MVID\_MINDBOX\_DYNAMICALLY': 'true',

'MVID\_NEW\_LK\_CHECK\_CAPTCHA': 'true',

'MVID\_NEW\_LK\_OTP\_TIMER': 'true',

'MVID\_NEW\_MBONUS\_BLOCK': 'true',

'MVID\_PODELI\_PDP': 'true',

'MVID\_SERVICES': '111',

'MVID\_SERVICE\_AVLB': 'true',

'MVID\_SINGLE\_CHECKOUT': 'true',

'MVID\_SP': 'true',

'MVID\_TYP\_CHAT': 'true',

'MVID\_WEB\_SBP': 'true',

'SENTRY\_ERRORS\_RATE': '0.1',

'SENTRY\_TRANSACTIONS\_RATE': '0.5',

'MVID\_GEOLOCATION\_NEEDED': 'true',

'\_\_lhash\_': 'eefc75736a01a8841a8afc69ced97529',

}

headers = {

'Accept': 'application/json',

'Connection': 'keep-alive',

'Accept-Language': 'ru',

'Host': 'www.mvideo.ru',

'User-Agent': 'Mozilla/5.0 (Macintosh; Intel Mac OS X 10\_15\_7) AppleWebKit/605.1.15 (KHTML, like Gecko) Version/15.6 Safari/605.1.15',

'Referer': 'https://www.mvideo.ru/noutbuki-planshety-komputery-8/noutbuki-118/f/tolko-v-nalichii=da?from=under\_search&showCount=72&page=7',

'x-set-application-id': '96595836-62d2-41b4-b2df-3d1ab4e39d28',

'baggage': 'sentry-environment=production,sentry-public\_key=1e9efdeb57cf4127af3f903ec9db1466,sentry-trace\_id=2d9bc6f65c1e4500b9b0af53d346d81c,sentry-sample\_rate=0.5,sentry-transaction=%2F\*\*%2F,sentry-sampled=true',

'sentry-trace': '2d9bc6f65c1e4500b9b0af53d346d81c-9ba5e42d08e8e729-1',

}

params = {

'categoryId': '118',

'offset': '432',

'limit': '72',

'filterParams': 'WyJ0b2xrby12LW5hbGljaGlpIiwiIiwiZGEiXQ==',

'doTranslit': 'true',

}

params1 = {

'categoryId': '118',

'offset': '0',

'limit': '72',

'filterParams': 'WyJ0b2xrby12LW5hbGljaGlpIiwiIiwiZGEiXQ==',

'doTranslit': 'true',

}

params2 = {

'categoryId': '118',

'offset': '72',

'limit': '72',

'filterParams': 'WyJ0b2xrby12LW5hbGljaGlpIiwiIiwiZGEiXQ==',

'doTranslit': 'true',

}

params3 = {

'categoryId': '118',

'offset': '144',

'limit': '72',

'filterParams': 'WyJ0b2xrby12LW5hbGljaGlpIiwiIiwiZGEiXQ==',

'doTranslit': 'true',

}

params4 = {

'categoryId': '118',

'offset': '216',

'limit': '72',

'filterParams': 'WyJ0b2xrby12LW5hbGljaGlpIiwiIiwiZGEiXQ==',

'doTranslit': 'true',

}

params5 = {

'categoryId': '118',

'offset': '288',

'limit': '72',

'filterParams': 'WyJ0b2xrby12LW5hbGljaGlpIiwiIiwiZGEiXQ==',

'doTranslit': 'true',

}

params6 = {

'categoryId': '118',

'offset': '360',

'limit': '72',

'filterParams': 'WyJ0b2xrby12LW5hbGljaGlpIiwiIiwiZGEiXQ==',

'doTranslit': 'true',

}

response = requests.get('https://www.mvideo.ru/bff/products/listing',

*params*=params, *cookies*=cookies, *headers*=headers).json()

total\_items = response.get('body').get('total')

if total\_items is None:

return '[!] No items :('

pages\_count = 6 # math.ceil(total\_items / 72)

print(*f*'[INFO] Total positions: {total\_items} | Total pages: {pages\_count}')

products\_ids = {}

products\_description = {}

products\_prices = {}

for i in range(pages\_count):

offset = *f*'{i \* 72}'

params = {

'categoryId': '118',

'offset': offset,

'limit': '72',

'filterParams': 'WyJ0b2xrby12LW5hbGljaGlpIiwiIiwiZGEiXQ==',

'doTranslit': 'true',

}

response = requests.get('https://www.mvideo.ru/bff/products/listing',

*params*=params, *cookies*=cookies, *headers*=headers).json()

products\_ids\_list = response.get('body').get('products')

products\_ids[i] = products\_ids\_list

products\_ids\_str = ','.join(products\_ids\_list)

params = {

'productIds': products\_ids\_str,

'addBonusRubles': 'true',

'isPromoApplied': 'true',

}

response = requests.get('https://www.mvideo.ru/bff/products/prices',

*params*=params, *cookies*=cookies, *headers*=headers).json()

material\_prices = response.get('body').get('materialPrices')

for item in material\_prices:

item\_id = item.get('price').get('productId')

item\_base\_price = item.get('price').get('basePrice')

item\_sale\_price = item.get('price').get('salePrice')

products\_prices[item\_id] = {

'item\_id': item\_id,

'item\_basePrice': item\_base\_price,

'item\_salePrice': item\_sale\_price

}

print(*f*'[+] Finished {i + 1} of the {pages\_count} pages')

response1 = requests.get('https://www.mvideo.ru/bff/products/listing',

*params*=params1, *cookies*=cookies, *headers*=headers).json()

response2 = requests.get('https://www.mvideo.ru/bff/products/listing',

*params*=params2, *cookies*=cookies, *headers*=headers).json()

response3 = requests.get('https://www.mvideo.ru/bff/products/listing',

*params*=params3, *cookies*=cookies, *headers*=headers).json()

response4 = requests.get('https://www.mvideo.ru/bff/products/listing',

*params*=params4, *cookies*=cookies, *headers*=headers).json()

response5 = requests.get('https://www.mvideo.ru/bff/products/listing',

*params*=params5, *cookies*=cookies, *headers*=headers).json()

response6 = requests.get('https://www.mvideo.ru/bff/products/listing',

*params*=params6, *cookies*=cookies, *headers*=headers).json()

# print(response)

products\_ids1 = response1.get('body').get('products')

products\_ids2 = response2.get('body').get('products')

products\_ids3 = response3.get('body').get('products')

products\_ids4 = response4.get('body').get('products')

products\_ids5 = response5.get('body').get('products')

products\_ids6 = response6.get('body').get('products')

json\_data1 = {

'productIds': products\_ids1,

'mediaTypes': [

'images',

],

'category': True,

'status': True,

'brand': True,

'propertyTypes': [

'KEY',

],

'propertiesConfig': {

'propertiesPortionSize': 5,

},

'multioffer': False,

}

json\_data2 = {

'productIds': products\_ids2,

'mediaTypes': [

'images',

],

'category': True,

'status': True,

'brand': True,

'propertyTypes': [

'KEY',

],

'propertiesConfig': {

'propertiesPortionSize': 5,

},

'multioffer': False,

}

json\_data3 = {

'productIds': products\_ids3,

'mediaTypes': [

'images',

],

'category': True,

'status': True,

'brand': True,

'propertyTypes': [

'KEY',

],

'propertiesConfig': {

'propertiesPortionSize': 5,

},

'multioffer': False,

}

json\_data4 = {

'productIds': products\_ids4,

'mediaTypes': [

'images',

],

'category': True,

'status': True,

'brand': True,

'propertyTypes': [

'KEY',

],

'propertiesConfig': {

'propertiesPortionSize': 5,

},

'multioffer': False,

}

json\_data5 = {

'productIds': products\_ids5,

'mediaTypes': [

'images',

],

'category': True,

'status': True,

'brand': True,

'propertyTypes': [

'KEY',

],

'propertiesConfig': {

'propertiesPortionSize': 5,

},

'multioffer': False,

}

json\_data6 = {

'productIds': products\_ids6,

'mediaTypes': [

'images',

],

'category': True,

'status': True,

'brand': True,

'propertyTypes': [

'KEY',

],

'propertiesConfig': {

'propertiesPortionSize': 5,

},

'multioffer': False,

}

# PAGE 1

response = requests.post('https://www.mvideo.ru/bff/product-details/list',

*cookies*=cookies, *headers*=headers, *json*=json\_data1).json()

extracted = pd.json\_normalize(response['body']['products'])

properties\_with\_id = pd.json\_normalize(response['body']['products'],

*record\_path*='propertiesPortion',

*meta*='productId',

*record\_prefix*='propertiesPortion.',

*errors*="ignore")

extracted.drop(['propertiesPortion'], *axis*='columns', *inplace*=True)

items1 = extracted.merge(properties\_with\_id, *how*='outer')

# print(items1)

# PAGE 2

response = requests.post('https://www.mvideo.ru/bff/product-details/list',

*cookies*=cookies, *headers*=headers, *json*=json\_data2).json()

extracted = pd.json\_normalize(response['body']['products'])

properties\_with\_id = pd.json\_normalize(response['body']['products'],

*record\_path*='propertiesPortion',

*meta*='productId',

*record\_prefix*='propertiesPortion.',

*errors*="ignore")

extracted.drop(['propertiesPortion'], *axis*='columns', *inplace*=True)

items2 = extracted.merge(properties\_with\_id, *how*='outer')

# print(items2)

# PAGE 3

response = requests.post('https://www.mvideo.ru/bff/product-details/list',

*cookies*=cookies, *headers*=headers, *json*=json\_data3).json()

extracted = pd.json\_normalize(response['body']['products'])

properties\_with\_id = pd.json\_normalize(response['body']['products'],

*record\_path*='propertiesPortion',

*meta*='productId',

*record\_prefix*='propertiesPortion.',

*errors*="ignore")

extracted.drop(['propertiesPortion'], *axis*='columns', *inplace*=True)

items3 = extracted.merge(properties\_with\_id, *how*='outer')

# print(products\_description3)

# PAGE 4

response = requests.post('https://www.mvideo.ru/bff/product-details/list',

*cookies*=cookies, *headers*=headers, *json*=json\_data4).json()

extracted = pd.json\_normalize(response['body']['products'])

properties\_with\_id = pd.json\_normalize(response['body']['products'],

*record\_path*='propertiesPortion',

*meta*='productId',

*record\_prefix*='propertiesPortion.',

*errors*="ignore")

extracted.drop(['propertiesPortion'], *axis*='columns', *inplace*=True)

items4 = extracted.merge(properties\_with\_id, *how*='outer')

# print(products\_description4)

# PAGE 5

response = requests.post('https://www.mvideo.ru/bff/product-details/list',

*cookies*=cookies, *headers*=headers, *json*=json\_data5).json()

extracted = pd.json\_normalize(response['body']['products'])

properties\_with\_id = pd.json\_normalize(response['body']['products'],

*record\_path*='propertiesPortion',

*meta*='productId',

*record\_prefix*='propertiesPortion.',

*errors*="ignore")

extracted.drop(['propertiesPortion'], *axis*='columns', *inplace*=True)

items5 = extracted.merge(properties\_with\_id, *how*='outer')

# print(products\_description5)

# PAGE 6

response = requests.post('https://www.mvideo.ru/bff/product-details/list',

*cookies*=cookies, *headers*=headers, *json*=json\_data6).json()

extracted = pd.json\_normalize(response['body']['products'])

properties\_with\_id = pd.json\_normalize(response['body']['products'],

*record\_path*='propertiesPortion',

*meta*='productId',

*record\_prefix*='propertiesPortion.',

*errors*="ignore")

extracted.drop(['propertiesPortion'], *axis*='columns', *inplace*=True)

items6 = extracted.merge(properties\_with\_id, *how*='outer')

# print(products\_description6)

df1 = pd.concat([items1, items2, items3, items4, items5, items6])

df2 = pd.DataFrame(products\_prices)

df2 = df2.T

print(df1, df2)

# ПОЛОЖИТЬ КЛЮЧИ В Variables

obs\_endpoint\_url = "https://storage.yandexcloud.net/"

obs\_access\_key\_id = "YCAJEY3cV8mry0Z3UIuydlb1F"

obs\_secret\_access\_key = "YCNRtS64Acm7fXUe23ms-z9BGUnFfI0-R\_Te-pPA"

obs\_bucket\_target = "otus-project"

obs\_bucket\_target\_lob\_name = "mvideo"

session = boto3.session.Session()

s3 = session.client(

*service\_name* = 's3',

*endpoint\_url* = obs\_endpoint\_url,

*aws\_access\_key\_id* = obs\_access\_key\_id,

*aws\_secret\_access\_key* = obs\_secret\_access\_key)

load\_dttm = datetime.today().strftime('%Y%m%d%H%M%S')

csv\_buffer\_item = StringIO()

df1.to\_csv(csv\_buffer\_item, *sep*=';', *mode*='w', *index*=False, *encoding* = 'utf-8')

s3.put\_object(*Bucket*=obs\_bucket\_target,

*Key*='{}/{}/{}.csv'.format(obs\_bucket\_target\_lob\_name, "products", "items"),

*Body*=csv\_buffer\_item.getvalue(),

*StorageClass*='COLD')

csv\_buffer\_price = StringIO()

df2.to\_csv(csv\_buffer\_price, *sep*=';', *mode*='w', *index*=False, *encoding* = 'utf-8')

s3.put\_object(*Bucket*=obs\_bucket\_target,

*Key*='{}/{}/{}.csv'.format(obs\_bucket\_target\_lob\_name, "products", "prices"),

*Body*=csv\_buffer\_price.getvalue(),

*StorageClass*='COLD')

*def* upload\_data():

obs\_endpoint\_url = "https://storage.yandexcloud.net/"

obs\_access\_key\_id = "YCAJEY3cV8mry0Z3UIuydlb1F"

obs\_secret\_access\_key = "YCNRtS64Acm7fXUe23ms-z9BGUnFfI0-R\_Te-pPA"

obs\_bucket\_target = "otus-project"

obs\_bucket\_target\_lob\_name = "mvideo"

session = boto3.session.Session()

s3 = session.client(

*service\_name* = 's3',

*endpoint\_url* = obs\_endpoint\_url,

*aws\_access\_key\_id* = obs\_access\_key\_id,

*aws\_secret\_access\_key* = obs\_secret\_access\_key)

# object\_key = 'products.csv'

products = s3.get\_object(*Bucket*=obs\_bucket\_target,

*Key*='{}/{}/{}.csv'.format(obs\_bucket\_target\_lob\_name, "products", "items"))

body = products['Body']

csv\_string = body.read().decode('utf-8')

df1 = pd.read\_csv(StringIO(csv\_string), *sep*=';', *encoding* = 'utf-8')

prices = s3.get\_object(*Bucket*=obs\_bucket\_target,

*Key*='{}/{}/{}.csv'.format(obs\_bucket\_target\_lob\_name, "products", "prices"))

body = prices['Body']

csv\_string = body.read().decode('utf-8')

df2 = pd.read\_csv(StringIO(csv\_string), *sep*=';', *encoding* = 'utf-8')

print(df1, df2)

engine = create\_engine('postgresql://user1:admin12345@rc1a-5benbpvqoh86s0ms.mdb.yandexcloud.net:6432/db1')

df1.to\_sql(

'products',

engine,

*schema*='mvideo',

*if\_exists*='append',

*index*=False,

*dtype*={

"item\_stars": sqlalchemy.types.NUMERIC},

)

df2.to\_sql('prices', engine, *schema*='mvideo', *if\_exists*='append', *index*=False)

args = {

'owner': 'airflow',

'catchup': 'False',

}

with DAG(

*dag\_id*='pl\_mvideo\_api',

*default\_args*=args,

*schedule\_interval*='0 6 \* \* 2',

*start\_date*=days\_ago(2),

*max\_active\_runs*=1,

*tags*=['4otus', 'API'],

*catchup*=False,

) as dag:

task\_get\_data = PythonOperator(

*task\_id*='get\_data',

*python\_callable*=get\_data,

*provide\_context*=True,

*dag*=dag

)

task\_upload\_data = PythonOperator(

*task\_id*='upload\_data',

*python\_callable*=upload\_data,

*provide\_context*=True,

*dag*=dag

)

task\_get\_data >> task\_upload\_data